



***RSF Elektronik***



## USER MANUAL UFC 430

USB-INTERFACE MODULE

## TABLE OF CONTENTS

<b>1.</b>	<b>General Information .....</b>	<b>03</b>
1.1	Important Information .....	03
1.2	Application .....	03
1.3	Items supplied.....	03
<b>2.</b>	<b>Specifications .....</b>	<b>04</b>
2.1	Mechanical Design and Ambient Conditions .....	04
2.2	USB Bus .....	04
2.3	Counter Interface (X1, X2, X3) .....	04
2.4	I/O Interface (X4) .....	04
2.5	Counter Operating Modes.....	04
2.6	Latch Logic .....	05
<b>3.</b>	<b>Hardware .....</b>	<b>06</b>
3.1	Connecting Elements and "Ready" LED .....	06
3.2	Connector Pin Assignments .....	07
<b>4.</b>	<b>Description of the Functions .....</b>	<b>08</b>
4.1	Counter Interface.....	08
4.2	I/O Interface .....	10
4.2.1	Delay Timer for external Sync-In.....	11
4.2.2	DC Parameters .....	12
<b>5.</b>	<b>Instructions for Installation .....</b>	<b>13</b>
5.1	Installing the Hardware .....	13
5.2	Installing the Drivers .....	13
5.3	Installing the enclosed Demonstration Software .....	13
<b>6.</b>	<b>DLL Functions.....</b>	<b>14</b>
6.1	Overview of the DLL functions .....	14
6.1.1	General Functions.....	14
6.1.2	Functions for Counter and Encoder Mode.....	14
6.1.3	Functions for Error Messages and Status .....	15
6.1.4	Functions for load and clear Counter.....	15
6.1.5	Functions for latching and reading out Count Values.....	15
6.1.6	Functions for referencing .....	16
6.1.7	Functions for external Inputs and Outputs.....	16
6.1.8	Functions for the Timer .....	16
6.2	Reference of the DLL Functions.....	17
6.2.1	General Functions.....	17
6.2.2	Functions for Counter and Encoder Mode.....	19
6.2.3	Functions for Messages and Status.....	21
6.2.4	Functions for load and clear Counter.....	22
6.2.5	Functions for latching and reading out Count Values.....	23
6.2.6	Funcions for referencing.....	25
6.2.7	Functions for external Inputs and Outputs.....	27
6.2.8	Functions for the Timer .....	29
	List of Figures.....	30
	List of Tables .....	30
	History .....	31
	List of the DLL Functions.....	32

# 1. GENERAL INFORMATION

## 1.1 Important Information

- **Danger to components if these notes are not observed**
- **Please observe the safety precautions according to DIN EN 100 015 when handling ESD components (electrostatic discharge)**
- **Only use antistatic packaging material**
- **For mounting observe that the working place is properly grounded**
- **Do not engage or disengage any connectors while the power supply is switched on**

## 1.2 Application

The UFC 430 module serves to record and evaluate encoder signals. It can also be used as an event counter.

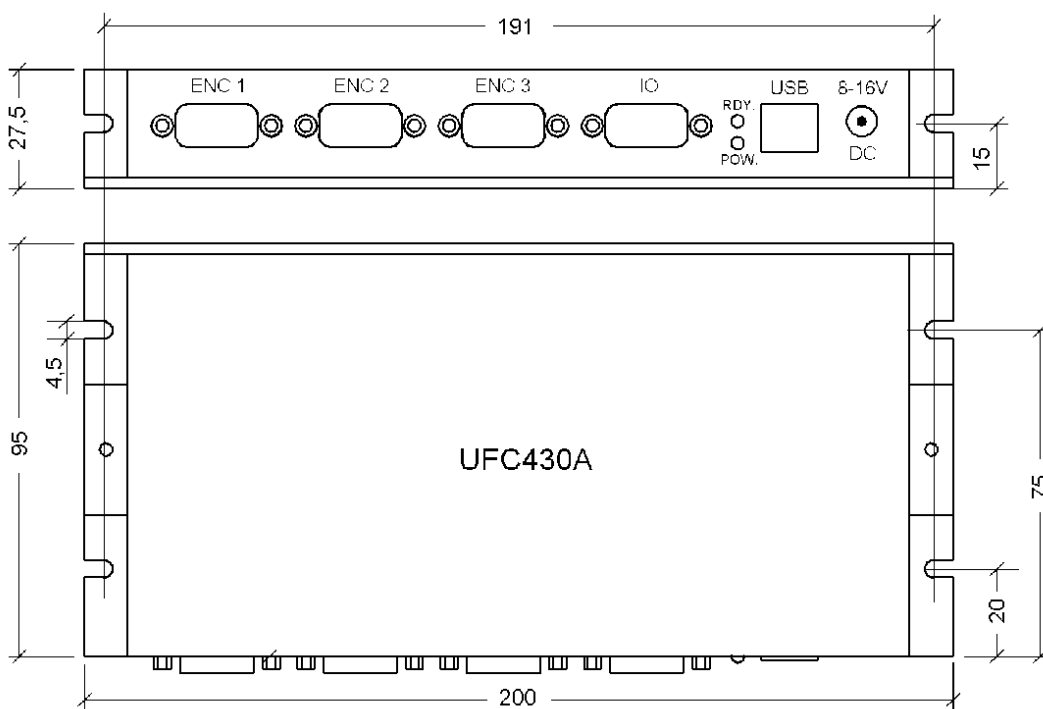
## 1.3 Items supplied

- UFC 430 (interface module)
- Wall power supply (option)
- CD with operating instructions, demo program and driver software

## 2. SPECIFICATIONS

### 2.1 Mechanical Design and Ambient Conditions

- Dimensions of the UFC 430 module: approx. 200 x 100 x 25 mm
- Maximum permissible ambient temperature: + 40 °C
- 3 D-Sub terminal strips, female, 15-pin (HD) for counter Inputs
- 1 D-Sub terminal strip, male, 15-pin (HD) for I/O signals
- USB connector, type B
- Low-voltage socket



### 2.2 USB Bus

- USB interface 2.0
- The functions of a USB bus are not described in this manual

## 2.3 Counter Interface (X1, X2, X3)

The definitions below apply for each of the encoder inputs!

- 3 RS422 (line driver) or analog Inputs (1 Vpp) for square-wave encoder signals and reference marks;
- Input frequency:
  - Encoders with TTL signals: max. 500 kHz ( $\Rightarrow$  slope time  $\geq 0.5 \mu\text{s}$ )
  - Encoders with analog signals (1Vpp):
    - max. 200 kHz at times 20 interpolation
    - max. 160 kHz at times 25 interpolation
    - max. 200 kHz at times 40 interpolation
    - max. 160 kHz at times 50 interpolation
    - max. 200 kHz at times 80 interpolation
    - max. 160 kHz at times 100 interpolation
    - max. 80 kHz at times 200 interpolation
    - max. 40 kHz at times 400 interpolation
- 1 TTL input for encoder interference signal
- 2 TTL inputs for encoder trigger signals
- Encoder power supply: 5.2 V; max. 0.2 A

## 2.4 I/O Interface (X4)

- 8 inputs (3–30 V) that can be used for ref. pulse inhibit, counter load signal, synchrones or asynchrones latch signal or for special assignment
- 4 outputs (TTL) to cascade several cards or for special assignment

## 2.5 Counter Operating Modes

- 3 counter channels, 32 bits each; one load and two latch registers for each channel
- To count the square-wave signals of encoders with TTL signals
- To count square-wave signals and additional interpolation of encoders with analog signals (1 Vpp)
- Event counter
- Integrated timer for pulse widths, frequency and speed measurements as well as cyclic storage of counter values in the latch registers

## 2.6 Latch Logic

- Asynchronous latching of the counter values for each encoder channel by software, reference mark of the encoder or external hardware signal
- Synchronous storage of several counters by software, timer or external signal
- Output signal for cascading several modules; can be programmed for software, timer or external hardware synchronisation
- Storing time: 62.5 ns

## 3. HARDWARE

### 3.1 Connecting Elements and "Ready" LED

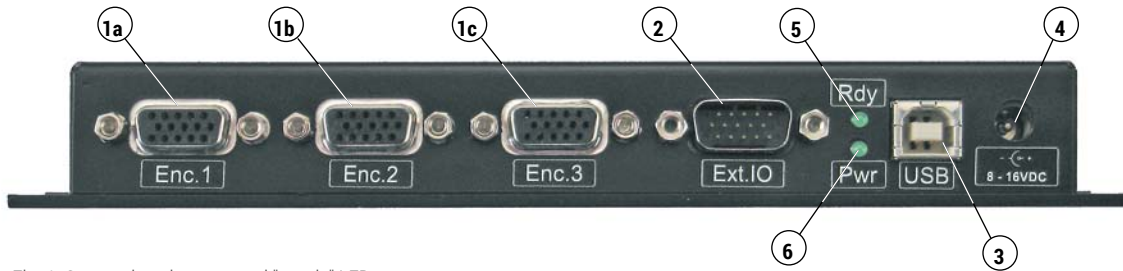


Fig. 1: Connecting elements and "Ready" LED

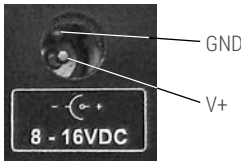
No.	Description
1a; 1b; 1c	X1; X2; X3 = D-Sub terminal strip, female, 15-pin for counter interface
2	X4 = D-Sub terminal strip, male, 15-pin for I/O interface
3	X5 = USB connector, type B
4	X6 = Low-voltage connector for external power supply
<b>Note!</b> 	

Table 1: Overview of the connecting elements

No.	Description
5	Rdy: Lights up as soon as the UFC430 module is ready for operation
6	Pwr: Lights up as soon as the external power is present

Table 2: Overview of the "Ready" LED

## 3.2 Connector Pin Assignments

Connector pin assignment X1, X2, X3:

Pin	Signal
1	Input A+
2	Input A-
3	Input B+
4	Input B-
5	Input R+
6	Input R-
7	Power supply +5V
8	GND
9	Shield
10	Trigger signal S1
11	Trigger signal S2
12	Power supply +5V sensor
13	GND sensor
14	Interference signal S-
15	not connected

Table 3: Connector pin assignment X1, X2, X3

Connector pin assignment X4:

Pin	Signal
1	IN 1
2	IN 2
3	IN 3
4	IN 4
5	IN 5
6	IN 6
7	IN 7
8	IN 8
9	Power supply +5V
10	GND
11	Out 1
12	Out 2
13	Out 3
14	Out 4
15	GND

Table 4: Connector pin assignment X4

## 4 DESCRIPTION OF THE FUNCTIONS

### 4.1 Counter Interface

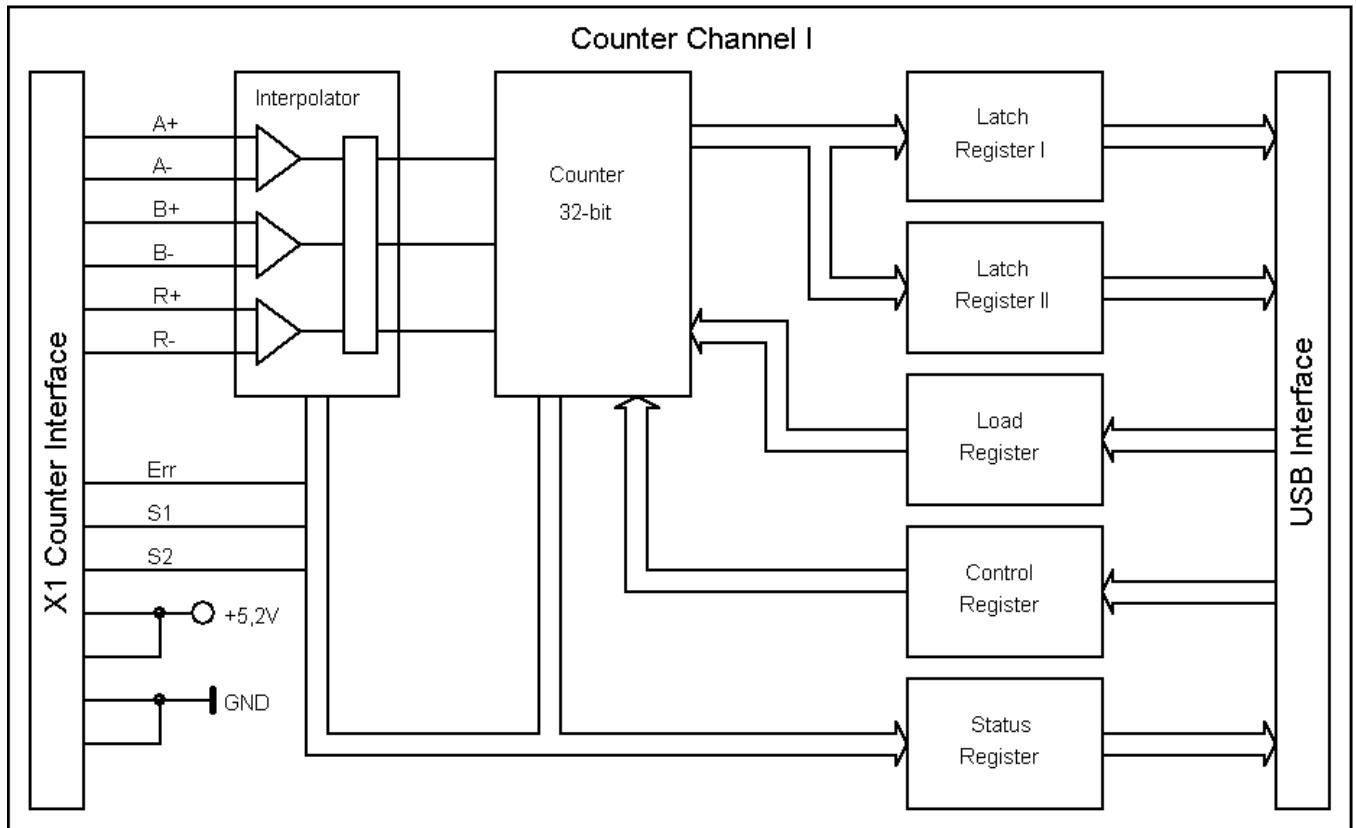


Fig. 2: Block diagram

The UFC 430 module features three equivalent counter channels (X1, X2, and X3) each with an interpolator, a counter, two latch registers, one load register, one control and one status register.

#### Counter operating modes

Three counter operating modes are available:

- Event counter with direction input and clear input
- Counting of square-wave signals with times 1, times 2 or times 4 evaluation for encoders with TTL signals
- Counting of square-wave signals with times 1, times 2 or times 4 evaluation and additional times 20, 25, 50 or times 100 interpolation can be programmed.

Used DLL function:

Name	Page	Function
UFC_SetInterpolMode	19	Switch between the three operating modes and simultaneously set the interpolation factor

Table 5: DLL functions, counter operating modes



### Load register

For each counter channel a 32-bit load register is available. The counter preset value must be written to the load register from where it is transferred to the counter by means of a software command or by a hardware event.

Used DLL functions:

Name	Page	Function
UFC_SetLoadReg	23	Write to load register
UFC_LoadCounter	23	Software command to transfer the value from the load register to the counter
UFC_SetLoadClearMode	22	Select a hardware source to transfer the contents of the load register to the counter

Table 6: DLL functions, load register

### Latch register

For each counter channel two latch registers are available. Before the count values can be read out, they must be stored in one of the latch registers. The values can be stored either individually for each counter channel or simultaneously for several counter channels, either by a software command or by a hardware event.

Used DLL functions:

Name	Page	Function
UFC_SetLatchMode	23	Select a hardware source to transfer the contents of the count value to the latch register
UFC_LatchImpuls	24	Generates a pulse that can be applied simultaneously to all latch registers and to the OUT 4 output (X4, pin 14)
UFC_LatchCounter	24	Software command to transfer a count value into a latch register

Table 7: DLL functions, latch register

### Status register

The following information can be obtained from the status register:

- Counter input signals (tracks A, B and R)
- Encoder interference signals (that may be present on encoders with TTL signals)
- Encoder amplitude monitoring (only active for encoders with 1 Vpp signals and integrated in the UFC 430 module)
- Encoder trigger signals (option for linear encoders)
- Monitoring of the encoder chain of steps
- Reference status (1st or 2nd reference mark traversed)

## 4.2 I/O Interface

The UFC430 module features an external I/O port (X4) with 8 inputs and 4 outputs. All inputs of the port are assigned special functions. If you do not need these special functions, the inputs are available to your requirements. The outputs of the port have no special functions (exception: OUT 4) and therefore they are always available. If OUT 4 is not required, it is also at your disposition.

The special functions of the external I/O port are listed in the table below:

PIN	Signal	Special function
1	IN 1	Ref. pulse inhibit for X1
2	IN 2	Asynchronous latch signal for X1
3	IN 3	Ref. pulse inhibit for X2
4	IN 4	Asynchronous latch signal for X2
5	IN 5	Ref. pulse inhibit for X3
6	IN 6	Asynchronous latch signal for X3
7	IN 7	Load signal for X1...X3
8	IN 8	Synchronous latch signal for X1...X3 (SyncIN)
11	OUT 1	Free
12	OUT 2	Free
13	OUT 3	Free
14	OUT 4	Cascading signal to cascade several modules (CascOUT)

Table 8: I/O functions

The following DLL functions are used to set the I/O functions:

Name	Page	Function
UFC_SetLoadClearMode	22	Select a hardware source to transfer the contents of the load register to the counter
UFC_SetRefInit	25	Activates ref. pulse inhibit
UFC_SetExtInit	27	Initializing of I/O port
UFC_SetLatchMode	23	Select a hardware source to transfer the contents of the count value to the latch register
UFC_SetExtOut	28	Sets the outputs

Table 9: DLL functions, external I/O port

### 4.2.1 Delay Timer for external Sync-In

Timer value (8-bit)	Timer
0	Timer off
1...255	Timer on

Table 10: Delay-Timer

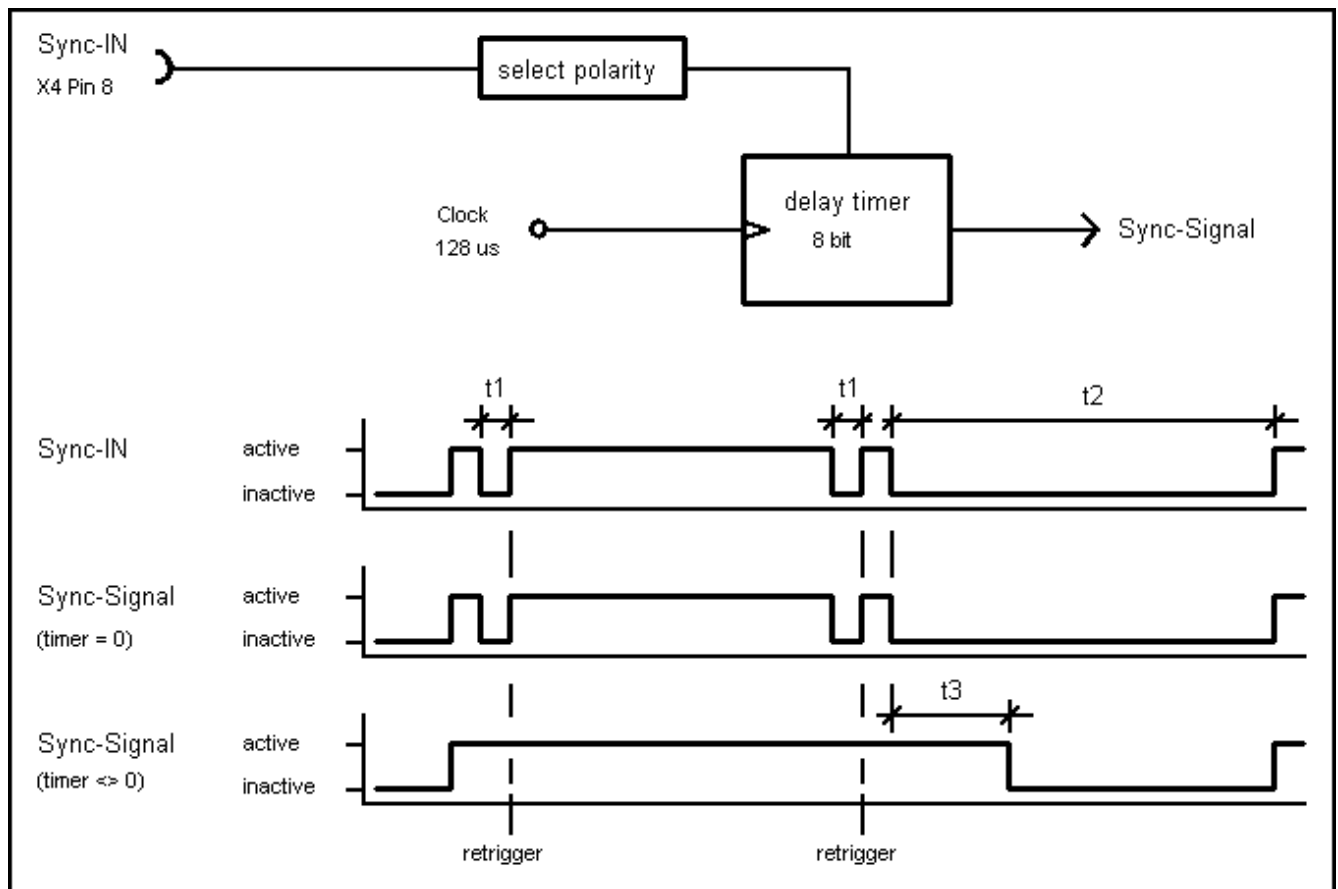


Fig. 3: Delay timer

$$t_1 < t_3 < t_2$$

$$t_3 = \text{timer value} \cdot 128 \mu\text{s}$$

The delay timer serves to activate a drop-out delay and thus a debouncing of the Sync-In input.

If the timer value is zero, the delay timer is inactive and the Sync signal directly follows the input.

If the timer value is " $\neq 0$ ", the timer is triggered each time the input signal is activated.

If the input signal ( $t_2$ ) is inactive longer than the timer ( $t_3$ ), the Sync signal is deactivated as soon as the timer has run off.

#### Notes!

The time  $t_3$  must be programmed longer than the time for the signal drops ( $t_1$ ).

If the CascOUT output (X4, pin 14) is programmed such that it is triggered in synchronism with SyncIN (X4, pin 8) (see DLL function "UFC\_SetExtInit" on page 27), the delay timer can be measured at CascOUT.

#### 4.2.2 DC Parameters

Parameter		Symbol	min	max	Unit
Input voltage	LOW	$V_{IL}$	0	0.7	V
	HIGH	$V_{IH}$	2.8	30	V
Output voltage	LOW	$V_{OL}$	-	1	V
	HIGH	$V_{OH}$	2.8	-	V
Output voltage	LOW	$I_{OL}$	-5	-	mA
	HIGH	$I_{OH}$	5	-	mA

Table 11: DC Parameters

## 5 INSTRUCTIONS FOR INSTALLATION

### 5.1 Installing the Hardware

**Before installation please observe the safety precautions according to DIN EN 100 015 when handling ESD components (electrostatic discharge)!**

- Connect the encoder and possibly available I/Os to the UFC 430 module. Before doing so make sure that you have disconnected the UFC 430 module from the main supply, since otherwise the encoders or the UFC 430 module may be damaged
- Connect the UFC 430 module to a personal computer using a USB cable
- Connect the UFC 430 module to the main supply via a wall power supply (12 VDC)

### 5.2 Installing the Drivers

#### **WINDOWS XP / VISTA / 7 / 8**

After the booting procedure the operating system automatically detects the UFC430 module. Now the appropriate drivers need to be installed.

- Insert the CD labelled "UFC430 Driver" into your CD-ROM drive
- Follow the instructions on the screen

After the drivers were successfully installed, an entry is made to the registry and the following files are copied to the system directory:

e.g. C:\WINDOWS\System32\Drivers\ UFC430.SYS

The enclosed disk contains also the DLLs.  
The corresponding DLL should be included directly in the application.

### 5.3 Installing the enclosed Demonstration Software

The demonstration program uses the previously installed drivers.  
No software installation is required here. You can start the demonstration program immediately from the CD-ROM or - after having copied it - from your computer hard disk.

## 6 DLL FUNCTIONS

### 6.1 Overview of the DLL Functions

#### 6.1.1 General Functions

Description	Brief reference	
Open device driver	integer UFC_OpenDrv	(void)
Close device driver	integer UFC_CloseDrv	(void)
Generate reset	integer UFC_SetReset	(unsigned char card)
Read out firmware version	integer UFC_GetFirmwareVersion	(unsigned char card, unsigned char *major, unsigned char *minor, unsigned char *version, unsigned char *revision)
Read out hardware version	integer UFC_GetHardwareVersion	(unsigned char card, unsigned long *Version)
Read out serial number (USB ID expansion)	integer UFC_GetSerialNumber	(unsigned char card, unsigned long *Number)
Read out external supply voltage	integer UFC_GetVoltExtern	(unsigned char card, unsigned short *Volt)
Read out encoder supply voltage	integer UFC_GetVoltEncoder	(unsigned char card, unsigned short *Volt)

Table 12: General functions

#### 6.1.2 Functions for Counter and Encoder Mode

Description	Brief reference	
Write interpolation factor	integer UFC_SetInterpolMode	(unsigned char module, unsigned char axis, unsigned char mode)
Read out interpolation factor	integer UFC_GetInterpolMode	(unsigned char card, unsigned char Axis, unsigned char *Mode)
Write counting direction	integer UFC_SetDirectionMode	(unsigned char card, unsigned char Axis, unsigned char Mode)
Read out counting direction	integer UFC_GetDirectionMode	(unsigned char card, unsigned char Axis, unsigned char *Mode)
Initialize encoder switch limit	integer UFC_SetSwitchInit	(unsigned char card, unsigned char Axis, unsigned char Init)
Read out encoder limit switch initialization	integer UFC_GetSwitchInit	(unsigned char card, unsigned char Axis, unsigned char *Init)

Table 13: Functions for counter and encoder mode

### 6.1.3 Functions for Error Messages and Status

Description	Brief reference	
Clear error messages	integer UFC_ClearError	(unsigned char card, unsigned char Clear)
Read out counter status	integer UFC_GetCounterStatus	(unsigned char card, unsigned char Axis, unsigned short *Status)

Table 14: Functions for error messages and status

### 6.1.4 Functions for load and clear Counter

Description	Brief reference	
Write counter load and clear mode	integer UFC_SetLoadClearMode	(unsigned char module, unsigned char axis, unsigned char mode)
Read out counter load and clear mode	integer UFC_GetLoadClearMode	(unsigned char card, unsigned char Axis, unsigned char *Mode)
Clear counter	integer UFC_ClearCounter	(unsigned char card, unsigned char Clear)
Load contents of load register to counter	integer UFC_LoadCounter	(unsigned char card, unsigned char Load)
Write load register	integer UFC_SetLoadReg	(unsigned char card, unsigned char Axis, unsigned long Data)

Table 15: Functions for load and clear counter

### 6.1.5 Functions for latching and reading out Count Values

Description	Brief reference	
Write latch register mode	integer UFC_SetLatchMode	(unsigned char module, unsigned char axis, unsigned char reg, unsigned char mode)
Read out latch register mode	integer UFC_GetLatchMode	(unsigned char card, unsigned char Axis, unsigned char Reg, unsigned char *Mode)
Generate latch pulse	integer UFC_LatchImpuls	(unsigned char card)
Copy count value to latch register	integer UFC_LatchCounter	(unsigned char card, unsigned char Latch)
Read out latch register	integer UFC_GetLatchReg	(unsigned char card, unsigned char Axis, unsigned char Reg, unsigned long *Data)

Table 16: Functions for latching and reading out count values

### 6.1.6 Functions for referencing

Description	Brief reference	
Initialize reference mode	integer UFC_SetRefINit	(unsigned char card, unsigned char Axis, unsigned char Init)
Read out reference mode	integer UFC_GetRefINit	(unsigned char card, unsigned char axis, unsigned char *Init)
Enable reference pulse(s)	integer UFC_ClearRef	(unsigned char card, unsigned char Clear)
Write to parameters for encoders with distance-coded referencemarks	integer UFC_RefPar	(unsigned char card, unsigned char Axis, unsigned short RefDis, unsigned short RefOffset, long Ref1, long Ref2, long PosOffset, long *EncOffset)
Read out count values in consideration of the distance-coded reference mark	integer UFC_GetPosRef	(unsigned char card, unsigned char Axis, unsigned long *Data)

Table 17: Functions for referencing

### 6.1.7 Functions for external Inputs and Outputs

Description	Brief reference	
Initialize mode of external inputs/outputs (X4)	integer UFC_SetExtINit	(unsigned char card, unsigned short Init)
Read out mode of external inputs/outputs (X4)	integer UFC_GetExtINit	(unsigned char card, unsigned short *Init)
Write external outputs (X4)	integer UFC_SetExtOUT	(unsigned char card, unsigned char Out)
Read out external inputs (X4)	integer UFC_GetExtIN	(unsigned char card, unsigned short *Input)

Table 18: Functions for external inputs and outputs

### 6.1.8 Functions for the Timer

Description	Brief reference	
Write timer value	integer UFC_SetTimer	(unsigned char card, unsigned short Timer)
Read out timer value	integer UFC_GetTimer	(unsigned char card, unsigned short *Timer)

Table 19: Functions for the timer



## 6.2 Reference of the DLL Functions

The DLL uses the following data types:

- unsigned char: 8 bits (no sign)
- unsigned char \*: Pointer to 8 bits (no sign)
- unsigned short: 16 bits (no sign)
- unsigned short \*: Pointer to 16 bits (no sign)
- integer: 16 bits (no sign)
- unsigned long: 32 bits (no sign)
- unsigned long \*: Pointer to 32 bits (no sign)

Every function returns a 16-bit integer:

**return = (integer)**

return = 0 ⇔ Data transfer faulty or module time-out

return = 1 ⇔ Data transfer successful

return = -1 ⇔ Driver not open

return = -2 ⇔ Driver still open but UFC430 module no longer connected

Up to 8 modules (cards 0-7) can be connected:

**card = (unsigned char)**

Each module features three counter inputs (axes 0–2).

**Axis = (unsigned char)**

### 6.2.1 General Functions

#### UFC\_OpenDrv

Open device driver

**Prototype:**     **return = UFC\_OpenDrv (void);**

return:         0 = Driver not found

                  1 = Driver opened

#### UFC\_CloseDrv

Close device driver

**Prototype:**     **return = UFC\_CloseDrv (void);**

return:         1 = Driver closed

#### UFC\_SetReset

Creates a software reset in the specified module

**Prototype:**     **return = UFC\_SetReset (unsigned char card);**

return:         (-2...1)

card:           Number of the module (0...7)

### UFC\_GetFirmwareVersion

Provides the firmware version of the UFC430 module

**Prototype:**     **return = UFC\_GetFirmwareVersion (unsigned char card, unsigned char \*major, unsigned char \*miNor, unsigned char \*version, unsigned char \*revision);**

return:           (-2...1)  
card:            Number of the module (0...7)  
\*major:          Firmware major  
\*miNor:          Firmware minor  
\*version:        Firmware version  
\*revision        Firmware revision

### UFC\_GetHardwareVersion

Provides the hardware version of the UFC430 module

**Prototype:**     **return = UFC\_GetHardwareVersion (unsigned char card, unsigned long \*Version);**

return:           (-2...1)  
card:            Number fo the module (0...7)  
\*Version:        Hardware version

### UFC\_GetSerialNumber

Provides the serial numer of the UFC430 module (USB ID expansion)

**Prototype:**     **return = UFC\_GetSerialNumber (unsigned char card, unsigned long \*Number);**

return:           (-2...1)  
card:            Number of the module (0...7)  
\*Number:         Serial number of the module

### UFC\_GetVoltExtern

Provides the voltage (read out external power supply)

**Prototype:**     **return = UFC\_GetVoltExtern (unsigned char card, unsigned short \*Volt);**

return:           (-2...1)  
card:            Number of the module (0...7)  
\*Volt:            Supply voltage of the module [0.1 V]

### UFC\_GetVoltEncoder

Provides the amplitude of the encoder supply voltage

**Prototype:**     **return = UFC\_GetVoltEncoder (unsigned char card, unsigned short \*Volt);**

return:           (-2...1)  
card:            Number of the module (0...7)  
\*Volt            Encoder supply voltage [0.1 V]

## 6.2.2 Functions for Counter and Encoder Mode

### UFC\_SetInterpolMode

Write interpolation factor

**Prototype:**     **return = UFC\_SetInterpolMode (unsigned char card, unsigned char Axis, unsigned char Mode);**

return:           (-2...1)

card:            Number of the module (0...7)

Axis:            Number of the axis (0...2)

Mode:            Counter operating mode (0-11)

operating mode 0:

Counter input without phase discriminator (event counter)

Track A = Counting-direction signal

Track B = Counter clock signal

Track R = Counter load or latch signal

operating mode 1-3:

Counter input with phase discriminator (for encoders with TTL signals only)

1 = times1 evaluation

2 = times2 evaluation

3 = times4 evaluation

operating mode 4-11:

Counter input with phase discriminator

(only for encoders with 1 Vpp signals)

4 = times20 interpolation

5 = times25 interpolation

6 = times40 interpolation

7 = times50 interpolation

8 = times80 interpolation

9 = times100 interpolation

10 = times200 interpolation

11 = times400 interpolation

### UFC\_GetInterpolMode

Read out interpolation factor

**Prototype:**     **return = UFC\_GetInterpolMode (unsigned char card, unsigned char Axis, unsigned char \*Mode);**

return:           (-2...1)

card:            Number of the module (0...7)

Axis:            Number of the axis (0...2)

\*Mode:           Selected counter operating mode (0-11)

### **UFC\_SetDirectionMode**

Write counting direction

**Prototype:**     **return = UFC\_SetDirectionMode (unsigned char card, unsigned char Axis, unsigned char Mode);**

return:           (-2...1)  
card:            Number of the module (0...7)  
Axis:            Number of the axis (0...2)  
Mode:            Counting direction  
                  0 = normal  
                  1 = inverted

### **UFC\_GetDirectionMode**

Read out counting direction

**Prototype:**     **return = UFC\_GetDirectionMode (unsigned char card, unsigned char Axis, unsigned char \*Mode);**

return:           (-2...1)  
card:            Number of the module (0...7)  
Axis:            Number of the axis (0...2)  
\*Mode:           Selected counting direction

### **UFC\_SetSwitchInit**

Initialization of the encoder limit switches

**Prototype:**     **return = UFC\_SetSwitchInit (unsigned char card, unsigned char Axis, unsigned char INit);**

return:           (-2...1)  
card:            Number of the module (0...7)  
Axis:            Number of the axis (0...2)  
INit:            0 ⇔ Encoder limit switch LO-active  
                  1 ⇔ Encoder limit switch HI-active

### **UFC\_GetSwitchInit**

Read out initialization of the encoder limit switches

**Prototype:**     **return = UFC\_GetSwitchInit (unsigned char card, unsigned char Axis, unsigned char \*INit);**

return:           (-2...1)  
card:            Number of the module (0...7)  
Axis:            Number of the axis (0...2)  
\*INit:           Setting of encoder limit switch (0 or 1)

## 6.2.3 Functions for Error Messages and Status

### UFC\_ClearError

Clear error messages (status bit 4)

**Prototype:**     **return = UFC\_ClearError (unsigned char card, unsigned char Clear);**

return:     (-2...1)

card:       Number of the module (0...7)

Clear:      Bit 0 = 0 ⇔ Do not clear error for counter 1 (axis 0)  
               1 ⇔ Clear error for counter 1 (axis 0)

Bit 1 = 0 ⇔ Do not clear error for counter 2 (axis 0)  
               1 ⇔ Clear error for counter 2 (axis 1)

Bit 2 = 0 ⇔ Do not clear error for counter 3 (axis 0)  
               1 ⇔ Clear error for counter 3 (axis 2)

### UFC\_GetCounterStatus

Read out counter status

**Prototype:**     **return = UFC\_GetCounterStatus (unsigned char card, unsigned char Axis, unsigned short \*Status);**

return:     (-2...1)

card:       Number of the module (0...7)

Axis:       Number of the axis (0...2)

\*Status:     Bit 0 = 0 ⇔ Encoder track A inactive  
               1 ⇔ Encoder track A active

Bit 1 = 0 ⇔ Encoder track B inactive  
               1 ⇔ Encoder track B active

Bit 2 = 0 ⇔ Encoder track R inactive  
               1 ⇔ Encoder track R active

Bit 3 = 0 ⇔ Encoder signal monitoring inactive  
               1 ⇔ Encoder signal monitoring active  
               (Encoder input pin 14 of encoders with TTL signals,  
               amplitude monitoring for encoders with analog input (1 Vpp))

Bit 4 = 0 ⇔ Encoder frequency monitoring inactive  
               1 ⇔ Encoder frequency monitoring active

Bit 5 = 0 ⇔ First reference mark not traversed  
               1 ⇔ First reference mark traversed

Bit 6 = 0 ⇔ Second reference mark not traversed  
               1 ⇔ Second reference mark traversed

Bit 7 = 0 ⇔ Reference pulse inhibit inactive  
               1 ⇔ Reference pulse inhibit active

Bit 8 = 0 ⇔ Encoder trigger signal S1 inactive  
               1 ⇔ Encoder trigger signal S1 active

Bit 9 = 0 ⇔ Encoder trigger signal S2 inactive  
               1 ⇔ Encoder trigger signal S2 active

## 6.2.4. Functions for load and clear Counter

### UFC\_SetLoadClearMode

Write counter load and clear mode

**Prototype:**     **return = UFC\_SetLoadClearMode (unsigned char card, unsigned char Axis, unsigned char Mode);**

return:           (-2...1)  
card:             Number of the module (0...7)  
Axis:             Number of the axis (0...2)  
Mode:             Counter load/clear mode (0–7) with hardware signal  
                    0 = Hardware signals locked  
                    1 = Clear counter with next encoder reference pulse  
                    2 = Clear counter all encoder reference pulses  
                    3 = Clear counter with integrated timer  
                    4 = Load counter with next encoder reference pulse  
                    5 = Load counter all encoder reference pulses  
                    6 = Clear counter with all encoder reference pulses and  
                       additionally load counter on negative zero crossover  
                    7 = Load counter with external signal (X4 pin 8)

### UFC\_GetLoadClearMode

Read out counter and clear mode

**Prototype:**     **return = UFC\_GetLoadClearMode (unsigned char card, unsigned char Axis, unsigned char \*Mode);**

return:           (-2...1)  
card:             Number of the module (0...7)  
Axis:             Number of the axis (0...2)  
\*Mode:            Selected counter load/clear mode (0–7)

### UFC\_ClearCounter

Clear counter

**Prototype:**     **return = UFC\_ClearCounter (unsigned char card, unsigned char Clear);**

return:           (-2...1)  
card:             Number of the module (0...7)  
Clear:            Bit 0 = 0 ⇒ Do not clear counter 1 (axis 0)  
                    1 ⇒ Clear counter 1 (axis 0)  
  
                    Bit 1 = 0 ⇒ Do not clear counter 2 (axis 1)  
                              1 ⇒ Clear counter 2 (axis 1)  
  
                    Bit 2 = 0 ⇒ Do not clear counter 3 (axis 2)  
                              1 ⇒ Clear counter 3 (axis 2)

## UFC\_LoadCounter

A separate load register is available for each counter channel. With this function the individual counters can be loaded with the contents of these registers. The counters can also be loaded by a number of hardware sources (see UFC\_SetLoadClearMode on page 22).

**Prototype:**     **return = UFC\_LoadCounter (unsigned char card, unsigned char Load);**  
 return:         (-2...1)  
 card:           Number of the module (0...7)  
 Load:           Bit 0 = 1 ⇨ Counter 1 (axis 0) is loaded with the contents of its load register  
                   Bit 1 = 1 ⇨ Counter 2 (axis 1) is loaded with the contents of its load register  
                   Bit 2 = 1 ⇨ Counter 3 (axis 2) is loaded with the contents of its load register

## UFC\_SetLoadReg

Write to load register

**Prototype:**     **return = UFC\_SetLoadReg (unsigned char card, unsigned char Axis, unsigned long Data);**  
 return:         (-2...1)  
 card:           Number of the module (0...7)  
 Axis:           Number of the axis (0...2)  
 Data:           32-Bit value for load register

## 6.2.5 Functions for latching and reading out count values

### UFC\_SetLatchMode

Write latch register mode

**Prototype:**     **return = UFC\_SetLatchMode (unsigned char card, unsigned char Axis, unsigned char Reg, unsigned char Mode);**  
 return:         (-2...1)  
 card:           Number of the module (0...7)  
 Axis:           Number of the axis (0...2)  
 Reg:            Latch register (0 or 1)  
 Mode:           Counter latch mode (0–7) with hardware signal  
                   0 = Hardware signals locked  
                   1 = Latch current count with software pulse (UFC\_LatchImpuls)  
                   2 = Latch current count with integral timer  
                   3 = Latch current count via external Sync-In at X4 (pIN 8)  
                   4 = Latch current count via external signal at X4  
                       IN 2 (Pin 2) for counter channel 1 (axis 0)  
                       IN 4 (Pin 4) for counter channel 2 (axis 1)  
                       IN 6 (Pin 6) for counter channel 3 (axis 2)  
                   5 = Latch current count with next encoder reference pulse  
                   6 = Latch current count with second encoder reference pulse  
                   7 = Latch current count with all encoder reference pulses

## UFC\_GetLatchMode

Read out latch register mode

**Prototype:**     **return = UFC\_GetLatchMode (unsigned char card, unsigned char Axis, unsigned char Reg, unsigned char \*Mode);**

return:           (-2...1)  
card:             Number of the module (0...7)  
Axis:             Number of the axis (0...2)  
Reg:              Latch register (0 or 1)  
\*Mode:            Selected counter latch mode (0-7)

## UFC\_LatchImpuls

Generate latch pulse

**Prototype:**     **return = UFC\_LatchImpuls (unsigned char card);**

return:           (-2...1)  
card:             Number of the module (0...7)

## UFC\_LatchCounter

For each counter channel two separate latch registers (reg. 0 and 1) are available.

With this function the individual count values can be loaded into the latch registers by software.

The count values can also be loaded into a latch register by a number of hardware sources (see UFC\_SetLatchMode on page 23).

**Prototype:**     **return = UFC\_LatchCounter (unsigned char card, unsigned char Latch);**

return:           (-2...1)  
card:             Number of the module (0...7)  
Latch:            Bit 0 = 1 ⇒ Store value of counter 1 (axis 0) in the corresponding latch reg. 0  
                    Bit 1 = 1 ⇒ Store value of counter 1 (axis 0) in the corresponding latch reg. 1  
                    Bit 2 = 1 ⇒ Store value of counter 2 (axis 1) in the corresponding latch reg. 0  
                    Bit 3 = 1 ⇒ Store value of counter 2 (axis 1) in the corresponding latch reg. 1  
                    Bit 4 = 1 ⇒ Store value of counter 3 (axis 2) in the corresponding latch reg. 0  
                    Bit 5 = 1 ⇒ Store value of counter 3 (axis 2) in the corresponding latch reg. 1

## UFC\_GetLatchReg

Read out latch register

**Prototype:**     **return = UFC\_GetLatchReg (unsigned char card, unsigned char Axis, unsigned char Reg, unsigned long \*Data);**

return:           (-2...1)  
card:             Number of the module (0...7)  
Axis:             Number of the axis (0...2)  
Reg:              Latch register (0 or 1)  
\*Data:            Contents of the latch register



## 6.2.6 Functions for referencing

### UFC\_SetRefInit

Initializing of external inputs as reference-pulse inhibitor.

**Prototype:**     **return = UFC\_SetRefInit (unsigned char card, unsigned char Axis, unsigned char Init);**  
 return:           (-2...1)  
 card:            Number of the module (0...7)  
 Axis:            Number of the axis (0...2)  
 Init:            0 ⇒ External input has no influence on the reference pulse  
                   1 ⇒ External input acts as reference pulse inhibitor  
                       IN 1 (pin 1) for counter channel 1 (axis 0)  
                       IN 3 (pin 3) for counter channel 2 (axis 1)  
                       IN 5 (pin 5) for counter channel 3 (axis 2)

### UFC\_GetRefInit

Read out initialization of the external inputs for ref. pulse inhibit

**Prototype:**     **return = UFC\_GetRefInit (unsigned char card, unsigned char Axis, unsigned char \*Init);**  
 return:           (-2...1)  
 card:            Number for the module (0...7)  
 Axis:            Number of the axis (0...2)  
 \*Init:           Setting for ref. pulse inhibit (0 or 1)

### UFC\_ClearRef

Ref. pulse enable (deletes the reference status)

**Prototype:**     **return = UFC\_ClearRef (unsigned char card, unsigned char Clear);**  
 return:           (-2...1)  
 card:            Number for the module (0...7)  
 Clear:           Bit 0 = 0 ⇒ Do not clear ref. status of counter 1 (axis 0)  
                   1 ⇒ Clear ref. status for counter 1 (axis 0)  
                   Bit 1 = 0 ⇒ Do not clear ref. status of counter 2 (axis 1)  
                   1 ⇒ Clear ref. status for counter 2 (axis 1)  
                   Bit 2 = 0 ⇒ Do not clear ref. status of counter 3 (axis 2)  
                   1 ⇒ Clear ref. status for counter 3 (axis 2)

## UFC\_RefPar

Write to parameters for encoders with distance-coded reference marks

**Prototype:**     **return = UFC\_RefPar (unsigned char card, unsigned char Axis, unsigned short RefDis, unsigned short RefOffset, long Ref1, long Ref2, long PosOffset, long \*EncOffset);**

return:           (-2...1)  
card:            Number of the module (0...7)  
Axis:            Number of the axis (0...2)  
RefDis:          Basic spacing of the reference marks  
RefOffset:       Reference offset  
Ref1:            Current count on first reference mark  
Ref2:            Current count on second reference mark  
PosOffset:       Counter offset  
\*EncOffset:      Encoder offset

## UFC\_GetPosRef

Read out count values in consideration of the distance-coded reference mark

**Prototype:**     **return = UFC\_GetPosRef (unsigned char card, unsigned char Axis, unsigned char Reg, unsigned long \*Data);**

return:           (-2...1)  
card:            Number of the module (0...7)  
Axis:            Number of the axis (0...2)  
Reg:             Latch register (0 or 1)  
\*Data:           Contents of latch register set against the reference parameters

## 6.2.7 Functions for external Inputs and Outputs

### UFC\_SetExtInit

Initialization of the polarity for the external inputs and outputs at X4

**Prototype:**     **return = UFC\_SetExtInit (unsigned char card, unsigned short Init);**  
 return:           (-2...1)  
 card:            Number of the module (0...7)  
 Init:            Bit 0 = 0 ⇔    Output 4 (Casc-Out) not triggered by software latch impuls  
                   1 ⇔    Output 4 (Casc-Out) triggered by software latch pulse  
  
                   Bit 1 = 0 ⇔    Output 4 (Casc-Out) not triggered by internal timer  
                   1 ⇔    Output 4 (Casc-Out) triggered by internal timer  
  
                   Bit 2 = 0 ⇔    Output 4 (Casc-Out) not triggered X4/pin 8 (Sync-In)  
                   1 ⇔    Output 4 (Casc-Out) triggered by X4/pin 8 (Sync-In)  
  
                   Bit 3 = 0 ⇔    Output 4 (Casc-Out) LO-active  
                   1 ⇔    Output 4 (Casc-Out) HI-active  
  
                   Bit 4 = 0 ⇔    Input 1 LO-active  
                   1 ⇔    Input 1 HI-active  
  
                   Bit 5 = 0 ⇔    Input 2 LO-active  
                   1 ⇔    Input 2 HI-active  
  
                   Bit 6 = 0 ⇔    Input 3 LO-active  
                   1 ⇔    Input 3 HI-active  
  
                   Bit 7 = 0 ⇔    Input 4 LO-active  
                   1 ⇔    Input 4 HI-active  
  
                   Bit 8 = 0 ⇔    Input 5 LO-active  
                   1 ⇔    Input 5 HI-active  
  
                   Bit 9 = 0 ⇔    Input 6 LO-active  
                   1 ⇔    Input 6 HI-active  
  
                   Bit 10 = 0 ⇔   Input 7 LO-active  
                   1 ⇔    Input 7 HI-active  
  
                   Bit 11 = 0 ⇔   Input 8 LO-active  
                   1 ⇔    Input 8 HI-active

### UFC\_GetExtInit

Read out initialization of the polarities for the external inputs and outputs at X4

**Prototype:**     **return = UFC\_GetExtInit (unsigned char card, unsigned short \*Init);**  
 return:           (-2...1)  
 card:            Number of the module (0...7)  
 \*Init:           Setting the polarities (0...65535)

## UFC\_SetExtOut

Write external outputs (X4)

**Prototype:**     **return = UFC\_SetExtOut (unsigned char card, unsigned char Out);**

return:        (-2...1)

card:          Number fo the module (0...7)

OUT:          Bit 0 = 0 ⇨   Output 1 inactive  
              1 ⇨   Output 1 active

              Bit 1 = 0 ⇨   Output 2 inactive  
              1 ⇨   Output 2 active

              Bit 2 = 0 ⇨   Output 3 inactive  
              1 ⇨   Output 3 active

              Bit 3 = 0 ⇨   Output 4 inactive  
              1 ⇨   Output 4 active

## UFC\_GetExtIn

Read out external inputs (X4)

**Prototype:**     **return = UFC\_GetExtIn (unsigned char card, unsigned short \*Input);**

return:        (-2...1)

card:          Number fo the module (0...7)

\*Input:        Bit 0 = 0 ⇨   Input 1 inactive  
              1 ⇨   Input 1 active

              Bit 1 = 0 ⇨   Input 2 inactive  
              1 ⇨   Input 2 active

              Bit 2 = 0 ⇨   Input 3 inactive  
              1 ⇨   Input 3 active

              Bit 3 = 0 ⇨   Input 4 inactive  
              1 ⇨   Input 4 active

              Bit 4 = 0 ⇨   Input 5 inactive  
              1 ⇨   Input 5 active

              Bit 5 = 0 ⇨   Input 6 inactive  
              1 ⇨   Input 6 active

              Bit 6 = 0 ⇨   Input 7 inactive  
              1 ⇨   Input 7 active

              Bit 7 = 0 ⇨   Input 8 inactive  
              1 ⇨   Input 8 active

### 6.2.8 Functions for the Timer

#### UFC\_SetTimer

Sets the preload value for the timer. If the preload value is set to "0", a running timer is stopped at the next zero crossover. If the preload value is  $\neq 0$ , a non-running timer is immediately started with the specified value; a running timer receives the new value at the next zero crossover.

$\text{Time} = (\text{Preload value} + 1) * 5\mu\text{s}$

**Prototype:**     **return = UFC\_SetTimer (unsigned char card, unsigned short Timer);**

return:     (-2...1)

card:       Number fo the module (0...7)

Timer:      Preload value (0...65535)

#### UFC\_GetTimer

Returns the run time remaining until the next zero crossover of the timer. During the zero crossover of the timer a signal is generated that can be used for a variety of functions depending on the initialization e.g. synchronous latching of count values, load counter or generating a pulse at the CascOut output.

**Prototype:**     **return = UFC\_GetTimer (unsigned char card, unsigned short \*Timer);**

return:     (-2...1)

card:       Number fo the module (0...7)

\*Timer      Remaining the time (0...65535)

## LIST OF FIGURES

Fig. 1: Connecting elements and "Ready" LED.....	6
Fig. 2: Block diagram .....	8
Fig. 3: Delay timer.....	11

## LIST OF TABLES

Table 1: Overview of the connecting elements.....	6
Table 2: Overview of the "Ready" LED .....	6
Table 3: Connector pin assignment X1; X2; X3.....	7
Table 4: Connector pin assignment X4.....	7
Table 5: DLL functions, counter operating modes .....	8
Table 6: DLL functions, load register.....	9
Table 7: DLL functions, latch register .....	9
Table 8: I/O functions.....	10
Table 9: DLL functions, external I/O port.....	10
Table 10: Delay timer .....	11
Table 11: DC Parameters.....	12
Table 12: General functions .....	14
Table 13: Functions for counter and encoder mode.....	14
Table 14: Functions for error messages and status .....	15
Table 15: Functions for load and clear counter .....	15
Table 16: Functions latching and reading out count values .....	15
Table 17: Functions for referencing .....	16
Table 18: Functions for external inputs and outputs .....	16
Table 19: Functions for the timer .....	16

## HISTORY

Date	Revision	Seite
10/2006	Revision of the user manual 05/2006 - 12/2006 <ul style="list-style-type: none"><li>▪ accurate listing of input frequencys for encoders with TTL signals and analog signals (1Vpp)</li><li>▪ corrected the formular for the timer</li></ul>	05 29
01/2007	Revision of the user manual 12/2006 - 01/2007 <ul style="list-style-type: none"><li>▪ corrected input frequency for encoders with analog signals (1Vpp)</li></ul>	05
02/2013	Revision of the user manual 01/2007 - 02/2013 <ul style="list-style-type: none"><li>▪ insert table of history</li><li>▪ insert technical drawing of UFC 430-controller housing</li><li>▪ update of the installation instructions</li></ul>	31 04 13

## LIST OF THE DLL FUNCTIONS

UFC_ClearCounter .....	15, 22	UFC_GetVoltEncoder .....	14, 18
UFC_ClearError .....	15, 21	UFC_GetVoltExtern .....	14, 18
UFC_ClearRef .....	16, 25	UFC_LatchCounter .....	15, 24
UFC_CloseDrv .....	14, 17	UFC_LatchImpuls .....	15, 24
UFC_GetCounterStatus .....	15, 21	UFC_LoadCounter .....	15, 23
UFC_GetDirectionMode .....	14, 20	UFC_OpenDrv .....	14, 17
UFC_GetExtI .....	16, 28	UFC_RefPar .....	16, 26
UFC_GetExtInit .....	16, 27	UFC_SetDirectionMode .....	14, 20
UFC_GetFirmwareVersion .....	14, 18	UFC_SetExtInit .....	16, 27
UFC_GetHardwareVersion .....	14, 18	UFC_SetExtOut .....	16, 28
UFC_GetInterpolMode .....	14, 19	UFC_SetInterpolMode .....	14, 19
UFC_GetLatchMode .....	15, 24	UFC_SetLatchMode .....	15, 23
UFC_GetLatchReg .....	15, 24	UFC_SetLoadClearMode .....	15, 22
UFC_GetLoadClearMode .....	15, 22	UFC_SetLoadReg .....	15, 23
UFC_GetPosRef .....	16, 26	UFC_SetRefInit .....	16, 25
UFC_GetRefInit .....	16, 25	UFC_SetReset .....	14, 17
UFC_GetSerialNumber .....	14, 18	UFC_SetSwitchInit .....	14, 20
UFC_GetSwitchInit .....	14, 20	UFC_SetTimer .....	16, 29
UFC_GetTimer .....	16, 29		





# DISTRIBUTION CONTACTS

AUSTRIA <i>Corporate Head Quarters</i>	RSF Elektronik Ges.m.b.H.	A-5121 Tarsdorf 93	☎ +43 62 78 81 92-0 FAX +43 62 78 81 92-79	e-mail: info@rsf.at internet: www.rsf.at
FRANCE	HEIDENHAIN FRANCE sarl	2 Avenue de la Christallerie 92310 Sèvres	☎ +33 1 41 14 30 00 FAX +33 1 41 14 30 30	e-mail: info@heidenhain.fr internet: www.heidenhain.fr
GREAT BRITAIN	HEIDENHAIN (GB) Ltd.	200 London Road Burgess Hill West Sussex RH15 9RD	☎ +44 1444 247711 FAX +44 1444 870024	e-mail: sales@heidenhain.co.uk internet: www.heidenhain.co.uk
ITALY	HEIDENHAIN ITALIANA S.r.l.	Via Asiago, 14 20128 Milano (MI)	☎ +39 02 27075-1 FAX +39 02 27075-210	e-mail: info@heidenhain.it internet: www.heidenhain.it
NETHERLANDS	HEIDENHAIN NEDERLAND B.V.	Copernicuslaan 34 6710 BB EDE	☎ +31 318 58 18 00 FAX +31 318 58 18 70	e-mail: verkoop@heidenhain.nl internet: www.heidenhain.nl
SWEDEN	HEIDENHAIN Scandinavia AB	Storsätragränd 5 SE-12739 Skärholmen	☎ +46 8 531 933 50 FAX +46 8 531 933 77	e-mail: sales@heidenhain.se internet: www.heidenhain.se
SWITZERLAND	RSF Elektronik (Schweiz) AG	Vierstrasse 14 CH-8603 Schwerzenbach	☎ +41 44 955 10 50 FAX +41 44 955 10 51	e-mail: info@rsf.ch internet: www.rsf.ch
CHINA	RSF Elektronik	Tian Wei San Jie, Area A, Beijing Tianzhu Airport Industrial Zone Shunyi District, 101312 Beijing P.R. China	☎ +86 10 80 42 02 88 FAX +86 10 80 42 02 90	e-mail: cao.shizhi@rsf.cn internet: www.rsf.cn
JAPAN	HEIDENHAIN K.K.	Hulic Kojimachi Bldg., 9F 3-2 Kojimachi, Chiyoda-ku Tokyo, 102-0083	☎ +81 3 3234 7781 FAX +81 3 3262 2539	e-mail: sales@heidenhain.co.jp internet: www.heidenhain.co.jp
KOREA	HEIDENHAIN LTD.	202 Namsung Plaza, 9th Ace Techno Tower, 130, Digital-Ro, Geumcheon-Gu, Seoul, Korea 153-782	☎ +82 2 20 28 74 30	e-mail: info@heidenhain.co.kr internet: www.rsf.co.kr
USA	HEIDENHAIN CORPORATION	333 East State Parkway Schaumburg, IL 60173-5337	☎ +1 847 490 11 91	e-mail: info@heidenhain.com internet: www.rsf.net

Date 02/2013 ■ Art.Nr.1063885-01 ■ Dok.Nr. D1063885-00-A-01 ■ Technical adjustments in reserve!



## RSF Elektronik

Ges.m.b.H.

Linear Encoders  
Cable Systems  
Precision Graduations  
Digital Readouts

Certified acc. to  
DIN EN ISO 9001  
DIN EN ISO 14001

